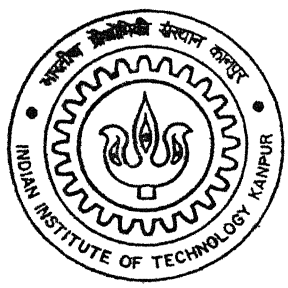
Solving Elliptic Partial Differential Equations using Classical Optimization methods and Genetic Algorithms

By

Saurabh Naik



DEPARTMENT OF MECHANICAL ENGINEERING

Indian Institute of Technology Kanpur

APRIL, 2002

Solving Elliptic Partial Differential Equations using Classical Optimization methods and Genetic Algorithms

A thesis submitted

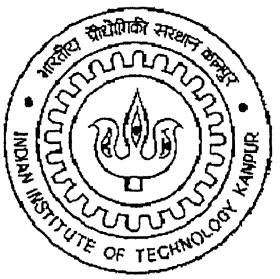
in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Saurabh Naik



to the

Department of Mechanical Engineering

Indian Institute of Technology
Kanpur-208016, India

April, 2002

Certificate

This is to certify that the work contained in the thesis entitled *Solving Elliptic Partial Differential Equations using classical Optmization methods and Genetic Algorithms* by *Saurabh Naik* is carried out under my supervision. This work has not been submitted elsewhere for a degree.

Kalyanmoy Deb
Professor
Department of Mechanical Engineering
Indian Institute of Technology
Kanpur -208016, India

T.K. Sengupta
Professor
Department of Aerospace Engineering
Indian Institute of Technology
Kanpur -208016, India

Dedicated to

My Parents

Acknowledgment

I express my sincere gratitude, regards and thanks to my supervisors Prof. Kalyanmoy Deb and Prof. T.K. Sengupta for excellent guidance, invaluable suggestions and generous help at all the stages of my research work. His interest and confidence in me was the reason for all the success I have made.

I would also like to thank Prof. B Sahay for helping me in overcoming some of the difficulties. I thank to my friends in the lab, Abhishek, Gulshan, Reddy, Sachin, Vivek and all others for their help and emotional support.

I feel indebted to Sudipta De for his invaluable suggestions regarding the CG method. I also acknowledge the help of all of my friends at Hall - IV, specially E - bottom, who made my stay at IIT-Kanpur memorable. I also thank all the people who have directly or indirectly helped in the completion of the thesis.

Saurabh Naik

Abstract

In many areas of physics and engineering elliptic partial differential occur. These equations are required to be solved numerically. Often large number of variables, complex geometry and boundary conditions are involved. Numerical solution of these equations consist in finite difference discretization of the continuous equation, and solving the resulting linear algebraic system. Instead of solving the linear system directly, classical methods solve it by minimizing the error in each iterations. This formulation suggest the use of Genetic Algorithms. GAs are optimization algorithms and are based on evolutionary principles of nature. GAs can be found better in the cases where the coefficient matrix in the algebraic system is asymmetric or the governing equation is non-linear. In the present study an algorithm is proposed which is a combination of classical methods and GA. This algorithm has been applied to $2 - D$ heat conduction problem and non-linear equation is also considered. Results are obtained for uniform as well as stretched grid, and are compared with the Conjugate gradient method. Problems where the coefficient matrix is asymmetric and in case of non-linear problems GAs are found better in terms of CPU time to reach a particular accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Numerical solution of partial differential equations (PDEs) in general, and *elliptic equations* in particular, is a challenge for the scientific computing community Problems in fluid mechanics, elasticity, heat transfer, electromagnetic theory, quantum theory and other areas of physics lead to elliptic equations. In-turn solving time dependent parabolic PDEs means solving elliptic equations at each time step. The most straightforward approach is to use a Taylor series expansion at different points in the domain and subsequent finite difference (FD) approximation. Boundary conditions are also dealt in the same fashion. One of the most important characteristics of elliptic PDEs is that the domain is closed and every point in the domain affects, and is affected by every other point in it [1]. Hence, the equation has to be simultaneously solved at each point in the domain. This leads to a large, sparse, and algebraic linear system but in general non-linear set of equations may also result. The whole problem of solving elliptic equations thus reduces to inverting the coefficient matrix. Almost all present day iterative methods rather than inverting the matrix, *minimizes* the error resulting from initial guess and further in each iteration. The minimization process is carried out either with the steepest-descent search or conjugate direction search. These methods are quite efficient and very popular. The early classical methods developed are due to Jacobi and Gauss-Seidel which are having convergence proofs [5] when the matrix satisfies certain properties. These methods do not solve the linear system as an optimization problem instead they take advantage of sparsity of the matrix and are based on LU decomposition. Conjugate gradient (CG) method developed five decades ago has become the most widely used method. Though applicable to symmetric positive definite systems after several modifications it can be applied to non symmetric systems as well. The variants of CG method are Bi conjugate gradient (BiCG), Conjugate gradient squared (CGS)

and Bi Conjugate gradient stabilized (BiCGSTAB). These methods come into more general category called *Krylov Subspace* methods [4].

The field of search and optimization has changed over last few years by the introduction of a number of non-classical, unorthodox and stochastic search and optimization algorithms. Of these, the Genetic Algorithms (GA) mimics nature's evolutionary principles to drive its search towards the optimum solution. One of the most striking differences of GA with the classical search and optimization algorithms is that GAs use a population of solutions in each iteration, instead of single solution. Since a population of solutions are processed in each iteration, the outcome of GA is also a population of solutions. If the optimization problem has a single optimum, all GA population members are expected to converge to that optimum solution. Another difference is that GA does not require restriction about the nature of the problem for its success, contrary to the classical methods. For example, CG method converges only for symmetric and positive definite system, and for Jacobi and Gauss-Seidel methods to converge, diagonal dominance is a must. Classical methods, except the Jacobi method are difficult, to parallelize, but GA being a population based approach, different members in the population can be processed by different processors. GAs have shown their ability in solving constrained, non-convex problems, and the problems where it is difficult to express the gradient with explicit mathematical function [20].

Objectives of present study

It is not aimed to present an algorithm, that will replace classical methods. The classical methods are, as they are, since they have been developed over centuries due to efforts of many researchers. Since classical methods tackle the problem of large sparse linear systems, that is $Ax = b$, as a minimization problem, it can also be tackled with GAs. Being a boundary value problem and the number of variables involved, makes the solution of large sparse linear algebriac systems even more challenging. The aim of the present work is to develop a GA that can solve such huge problems. Secondly to identify areas where it is worthwhile to apply GA, and finally to point out potential improvements needed in the present generation GA operators. Keeping these aim in mind, the algorithm developed has been applied to few problems of heat conduction. With such a narrow class of problems being tackled with, it is difficult to claim universality of the algorithm but, still this study presents a framework that can serve for further research in the area. In order to identify difficulties involved with classical methods theoretical aspects have been studied, involving Neumann and Dirichlet types of boundary conditions and their effect on complexity. Grid transformations have been applied when the grid is logarithmically stretched, for grid in arithmetic progression, equation has been solved in the physical plane itself. Non-linearity in the original Laplace's equation has been introduced by assuming the coefficient of thermal

conductivity as a linear function of temperature.

1.2 Organization of Thesis

- **Chapter 1** *Introduction*: This chapter summarizes the whole thesis and also discusses the objectives of this study
- **Chapter 2** *Overview of classical method and introduction to GAs*: In this chapter theoretical aspects of Gauss Seidel and CG method have been discussed also the operators used in GA have been described, finally the developed algorithm is presented.
- **Chapter 3** *Simulation results*: Simulation results are presented for different boundary conditions (BCs) and different types of grid such as arithmetic and logarithmic grid. Non-linear problem with coefficient thermal conductivity as a function of temperature is also solved.
- **Chapter 4** *Conclusion and further research*: Concluding remarks are given and suggestions for further work have also been included.

Chapter 2

Overview of classical methods and introduction to GAs

Although it is difficult to name any method as 'classical method' but this distinction has been made for those methods that follow a deterministic transition rule such as gradient based methods. These methods are point to point methods and in each iteration unidirectional search is performed, the direction itself is arrived at by using local information. In this chapter commonly used methods and those used for benchmarking has been presented. Some theoretical background of Jacobi, Gauss Seidel and CG method is also being given. At the end PCX operator that is used in GA is described and methodology of solving linear as well as nonlinear systems with GAs is presented

2.1 From PDEs to linear systems

In general elliptic PDEs are of the form

$$\frac{\partial}{\partial x}\left(f\frac{\partial \phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(g\frac{\partial \phi}{\partial y}\right)=F. \tag{2.1}$$

Where, f, g and F may be functions of space coordinates or of ϕ, in later case the equation will be non-linear. This equation is known as Piosson's equation. When $F=0$ and f and g are constants the equation reduces to Laplace's equation;

$$\nabla^2\phi=0. \quad . \tag{2.2}$$

For solving this equation central difference approximation of second derivative is used.

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\triangle x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\triangle y^2} = 0. \tag{2.3}$$

This difference equation is an approximation to the original equation (2.2) when the grid is uniform, that is, the node spacings are same in x or y direction. If spacing is same in x and y direction then this equation after transposing further reduces to,

$$\phi_{i,j} - \frac{1}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}) = 0. \tag{2.4}$$

For rectangular domain with M and N grid points on its two sides equation (2.4) generates a linear system with $M \times N$ equations. Since function value at any node in the domain is an average of the values at 4 neighboring points, the coefficient matrix of the linear system contains 5 diagonals, 2 are adjacent to the main diagonal and other two are positioned according to the values of M and N. The right side vector contains nonzero entries corresponding to nodes adjacent to the boundary. This system of equations in matrix form is written as $Ax = b$. The exact solution of this system of equations is not a solution of differential equation (2.2), but it is just an approximation accurate enough for all practical purposes. Further if $\triangle x \to 0$ and $\triangle y \to 0$ then the truncation error (neglecting the higher order terms in the FD approximation) goes to zero and the system of linear equations approaches to the original PDE. There are various methods to solve this system of equations, in fact the whole effort in solving the PDE lies in finding a solution. A brief description of these methods is given in the next section.

2.2 Classification

Methods of solution of general computational problems fall into two categories; *direct* and *indirect*. Direct methods, of which the solution of a tridiagonal system is typical are those which give exact answer to the system of algebraic equations in finite number of steps, if there were no round-off errors. The algorithm for such a procedure is often complicated and non-repetitive. Many direct methods for linear systems are available in literature [2, 3]. These methods have usually been omitted from consideration in past efforts because of excessive computer storage requirements, both in the program and in the necessity to store many intermediate results for later use. Direct methods have been adopted for banded matrices, and for solving medium-sized elliptic problems arising in finite element approximations Indirect or iterative methods are described next.

2.3 Iterative methods

The term *iterative methods* refers to a wide range of techniques, that use successive approximations to obtain more accurate solution to a linear system at each step. These methods are built around a partition of A in $Ax = b$.

$$A = L + D + U. \tag{2.5}$$

where D,$-L$ and $-U$ represent the diagonal, lower-triangular, and upper-triangular parts of A, respectively. Iterative methods in-turn can be classified into two sub categories namely *stationary* methods and *non-stationary* methods. Jacobi, Gauss-Seidel and Over-relaxation methods belong to stationary methods.

2.3.1 Jacobi Method

This scheme first used by Jacobi in 1844, is also called *iteration by total steps* and the *method of simultaneous displacements* can be easily derived by examining each of the N equations in the linear system in isolation. For i^{th} equation,

$$\sum_{j=1}^{n} a_{i,j}x_j = b_i, \tag{2.6}$$

It can be solved for the value of x_i while assuming that the other entries of x remain fixed. Thus we get,

$$x_i = (b_i - \sum_{j \neq i} a_{i,j}x_j)/a_{i,i}, \tag{2.7}$$

This equation can be converted into iterative scheme defined by;

$$x_i^{(k)} = (b_i - \sum_{j \neq i} a_{i,j}x_j^{(k-1)})/a_{i,i}, \tag{2.8}$$

which is the Jacobi method. Since the Jacobi method treats each equation independently the order in which elements of x are updated is immaterial. The Jacobi method can be expressed in matrix form [4];

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b, \tag{2.9}$$

if we replace $G = D^{-1}(L+U)$ and $M = D^{-1}$ then Jacobi method in compact form can be written as;

$$x^{(k)} = Gx^{(k-1)} + Mb. \tag{2 10}$$

The stationary nature of Jacobi method can be observed from the fact the matrices G and M remain same in each iteration. G can be expressed in terms of A as;

$$G = D^{-1}A - I. \tag{2.11}$$

One iteration of the Jacobi method takes $O(n)$ operations where n is the size of the matrix.

2.3.2 Gauss-Seidel Method

This iteration was used by Gauss in his calculations and was independently discovered by Seidel in 1874. Also known as method of *successive displacements* or *iterations by single steps*, is based upon *immediate* use of improved values. To carry out such a computation the order in which one solves for the components of the k^{th} approximation $x^{(k)}$ must be established beforehand. For an arbitrary but fixed lexicographic ordering, the Gauss-Seidel method is derivable from equation (2.8) by employing the improved values when available that is [4];

$$x^{(k)} = (b_i - \sum_{j<i} a_{i,j}x_j^{(k)} - \sum_{j>i} a_{i,j}x_j^{(k-1)})/a_{i,i}. \tag{2.12}$$

In matrix form, the Gauss-Seidel method can be expressed as:

$$x^{(k)} = (D+L)^{-1}(Ux^{(k-1)} + b). \tag{2.13}$$

Thus, the *Gauss-Seidel method is a linear stationary iterative scheme.*

2.3.3 Convergence of stationary methods

Every iterative procedure in general can be expressed as in equation (2.10) where $M_kA + G_k = I$. Convergence of the above iteration can be studied by examining an error vector

by

$$\begin{aligned} e_k &= x^{(k)} - A^{-1}b, \\ &= G_k x^{(k-1)} + M_k b - A^{-1}b, \\ &= G_k x^{(k-1)} + M_k b - G_k A^{-1}b - M_k b, \\ &= G_k e_{k-1}. \end{aligned} \tag{2.14}$$

Thus, e_k satisfies the basic iteration equation (??) with $b = 0$. Applying equation (??) repetitively

$$\begin{aligned} e_1 &= G_1 e_0, \\ e_2 &= G_2 G_1 e_0, \\ e_k &= G_k G_{k-1} \ldots G_1 e_0 \end{aligned}$$

Consequently, the convergence of the iteration, for a specified initial error e_0, depends upon whether

$$H_k e_0 = G_k G_{k-1} \ldots G_1 e_0 \to 0 \qquad as \qquad k \to \infty. \tag{2.15}$$

When the iteration is both stationary and linear, $G_k = G$ and $H_k = G^k$ so the convergence depends on magnitude of $G^k e_0$. Further it has been shown in [5], that the convergence depends upon the eigenvalues of G. We directly state the result;
"The stationary linear iteration $x^{(k)} = Gx^{(k-1)} + Mb$ converges if and only if the spectral radius of G is less than 1."
Where the spectral radius of a matrix is the magnitude of largest eigenvalue This theorem puts restriction on A. It is required that for Jacobi and Gauss-Seidel methods to converge, A should be diagonally dominant and this is a *sufficient* condition. The efficiency of these iterative methods is directly related to the spectral radius of the matrix G. The relative magnitudes of the spectral radii of the matrices associated with Jacobi and Gauss-Seidel methods have been examined by Stein and Rosenberg[8]. They obtained the following result;

If A satisfies the conditions;

$$(i) \qquad a_{i,i} > 0, \qquad a_{i,j} \leqq 0, i \neq j,$$

$$(ii) \qquad a_{i,j} \geq \sum_{j=1, i \neq j}^{N} |a_{i,j}|,$$

$$(iii) \qquad A \quad \text{is irreducible,}$$

and G_J, G_S are the matrices associated with the Jacobi and Gauss-Seidel iterations, respectively, then one and only one of the following mutually exclusive relations holds;

$$(i) \qquad \lambda(G_J) = \lambda(G_S) = 0;$$

$$(ii) \qquad 0 < \lambda(G_S) < \lambda(G_J) < 1;$$

$$(iii) \qquad 1 = \lambda(G_S) = \lambda(G_J);$$

$$(iv) \qquad 1 < \lambda(G_J) < \lambda(G_S).$$

Where the notation $\lambda(G)$ is used for spectral radius of matrix G. The content of this theorem is that the Jecobi and Gauss-Seidel methods are either both convergent or both divergent, and *if both converge then the Gauss-Seidel method converges faster.*

2.3.4 The conjugate gradient method

The CG method is one of the most popular non-stationary iterative method for symmetric and positive definite systems. Hestenes and Steifel [6] have shown the remarkable property that the correct solution is obtained after no more than n steps, where n is the number of equations. The CG method minimizes the error while searching in new direction *conjugate* to all previous directions. Here derivation of the CG method is presented [7] .
Consider the quadratic form:

$$f(x) = \frac{1}{2} x A^T x - b^T x + c, \tag{2.16}$$

Differentiating it with respect to x we get.

$$f^{'}(x) = Ax - b, \tag{2.17}$$

Following notations can be defined
$e^{(k)} = x^{(k)} - x$ is the error term at k^{th} iteration and x is the exact solution,

$r^{(k)} = b - Ax^{(k)}$ is the residual,
from these definitions we have the identities $-Ae^{(k)} = r^{(k)}$ and $r^{(k)} = f'(x^{(k)})$. In each step, the x vector is updated as follows.

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}. \tag{2.18}$$

The search directions are so chosen that they are A-orthogonal or A-conjugate that is:

$$d_i^T A d_j = 0. \tag{2.19}$$

The basic idea behind conjugate search direction is, that the search is carried out in a particular direction exactly once. This is ensured by taking $e^{(k+1)}$ A-orthogonal to $d^{(k)}$. This orthogonality is equivalent to finding the minimum point along the search direction $d^{(k)}$ [8].

$$\begin{aligned} \frac{d}{d\alpha} f(x^{(k+1)}) &= 0, \\ f'(x^{(k+1)}) \frac{d}{d\alpha} x^{(k+1)} &= 0, \\ -[r^{(k+1)}]^T d^{(k)} &= 0, \\ [d^{(k)}]^T A e^{(k+1)} &= 0, \end{aligned}$$

Substituting $e^{(k+1)}=e^{(k)} + \alpha^{(k)} d^{(k)}$ we get,

$$\alpha^{(k)} = \frac{[d^{(k)}]^T r^k}{[d^{(k)}]^T A d^{(k)}}.$$

To show that this procedure computes x in n steps, we express the error vector in terms of weighted sum of conjugate directions.

$$e^{(0)} = \sum_{j=0}^{n-1} \delta^{(j)} d^{(j)}. \tag{2.20}$$

Pre-multiplying the above expression by $[d^{(k)}]^T A$ and by using A-orthogonality of d- vectors, we obtain $\alpha^{(k)} = -\delta^{(k)}$. This result gives rise to following conclusion; the process of building up x-vector can also be viewed as cutting down the error term component by component Now the next step is to generate a set of A-orthogonal search directions, the most simple

way to do it is the *Gram-Schmidt process* [7]. Suppose we have n linearly independent vectors $u^{(0)}, u^{(1)}, ..., u^{(n-1)}$. The conjugate search directions are given by $d^{(0)} = u^{(0)}$ and for $k > 0$.

$$d^{(k)} = u^{(k)} + \sum_{i=0}^{k-1} \beta_{ki} d^{(i)}, \tag{2.21}$$

where β_{ki} is given by the following expression

$$\beta_{ki} = -\frac{[u^{(k)}]^T A d^{(i)}}{[d^{(i)}]^T A d^{(i)}}, \tag{2.22}$$

The difficulty in using the Gram-Schmidt conjugation in the CG method is that all the search vectors must be kept in memory to construct each new one, and furthermore $\mathcal{O}(n^3)$ operations are required to generate full set. In fact using Gram-Schmidt conjugation, becomes equivalent to performing Gauss elimination. But this difficulty is circumvented by utilizing a remarkable property, that each search direction is traversed only once so the error term (that is $e^{(k)}$) is A-orthogonal to all old search directions since $r^{(k)} = b - Ax^{(k)}$ the residue is also orthogonal to all search directions. Because the search directions are constructed from n (the size of the matrix) linearly independent set of vectors denoted by u, the space spanned by $u^{(0)}, u^{(1)}, ..., u^{(k-1)}$ is $\mathcal{D}^k$. The residual $r^{(k)}$ is orthogonal to all previous u vectors as well. This fact is employed in the CG method.The u vectors are chosen as the residuals themselves, hence the subspace span $r^{(0)}, r^{(1)}, ..., r^{(k-1)}$ is equal to $\mathcal{D}^k$. As each residual is orthogonal to previous search directions it is also orthogonal to previous residuals. Figure 2.1 shows this idea. The colored rigon is a $2 - D$ subspace spanned by the search directions $d^{(0)}$ and $d^{(1)}$ which are A-orthogonal. The residuals are orthogonal to each other, and they also belong to the same subspace spanned by search directions. The next search direction $d^{(2)}$ is A-orthogonal to previous search directions and the new residual $r^{(2)}$ is orthogonal to all previous residuals hence;

$$[r^{(k)}]^T r^{(i)} = 0, \qquad i \neq j. \tag{2.23}$$

Further the residual $r^{(k)}$ is just a linear combination of previous residuals and $Ad^{(k-1)}$. The fact $d^{(k-1)} \in \mathcal{D}^k$, implies that each new subspace $\mathcal{D}^k + 1$ is formed from the union of previous subspace $\mathcal{D}^k$ and the subspace $A\mathcal{D}^k$ hence,

$$\begin{aligned} \mathcal{D}^k &= span \quad \{d^{(0)}, Ad^{(0)}, A^2 d^{(0)}, \ldots, A^{(k-1)} d^{(0)}\}, \\ &= span \quad \{r^{(0)}, Ar^{(0)}, A^2 r^{(0)}, \ldots A^{(k-1)} r^{(0)}\}. \end{aligned}$$

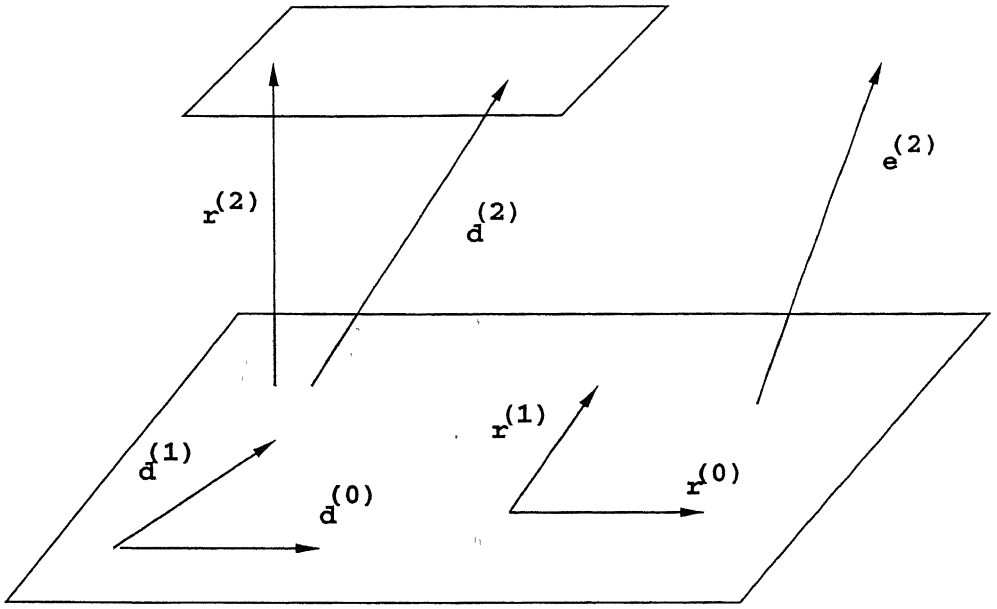


Figure 2.1: Shows each new residual is orthogonal to all previous residuals and search directions; and each new search direction is constructed from the residuals to be A-orthogonal to all previous residuals and search directions.

This subspace spanned by residuals is called *Krylov subspace*, a subspace created by repeatedly applying a matrix on a vector. Since $\mathcal{D}^k$ is included in $\mathcal{D}^{k+1}$, the fact that the next residual $r^{(k)}$ is orthogonal to $\mathcal{D}^{k+1}$ implies that $r^{(k+1)}$ is A-orthogonal to $\mathcal{D}^k$. The coefficient in the Gram-Schmidt conjugation (2.22) reduces to,

$$\beta^{(k)} = \frac{[r^{(k)}]^T r^{(k)}}{[r^{(k-1)}]^T r^{(k-1)}}. \tag{2.24}$$

Putting it all together we have the following algorithm for CG method [14] .

$$\begin{aligned}
u^{(k)} &= Ap^{(k)}, \\
\alpha^{(k)} &= \frac{[r^{(k)}]^T r^{(k)}}{[p^{(k)}]^T u^{(k)}}, \\
x^{(k+1)} &= x^{(k)} + \alpha^{(k)} p^{(k)}, \\
r^{(k+1)} &= r^{(k)} - \alpha^{(k)} p^{(k)}, \\
\beta^{(k+1)} &= \frac{[r^{(k+1)}]^T r^{(k+1)}}{[r^{(k)}]^T r^{(k)}}, \\
p^{(k+1)} &= r^{(k+1)} + \beta^{(k+1)} p^{(k)}.
\end{aligned}$$

with initial value for $k = 0$, and $p^{(0)} = r^{(0)} = b - Ax^{(0)}$. The initial trial vector $x^{(0)}$ is usually

taken to be null, unless some approximation to the solution is known from the previous similar analysis. Iterations are usually terminated when the residual is small. Since the initial residual is b if $x^{(0)} = 0$, a suitable criterion for halting the iterations is

$$\frac{||r^{(k)}||}{||b||} < \epsilon.$$

where ϵ is some pre-specified tolerance.

Convergence

Predictions about the convergence of the CG method is difficult to make, but suitable error bounds can be obtained in terms of $\kappa = \lambda(A)$ of the matrix $M^{-1}A$ when M is used as preconditioner. If $\hat{x}$ is the exact solution of the linear system $Ax = b$, with symmetric positive definite matrix A, then with symmetric positive definite preconditioner it can be shown that [10, 11]

$$||x^{(k)} - \hat{x}|| \leq 2\rho||x^{(0)} - \hat{x}||,$$

where $\rho = (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$. This relation shows that the number of iterations to reach a relative reduction of ϵ is proportional to $\sqrt{\kappa}$. Further if the extremal eigenvalues of the matrix $M^{-1}A$ are well separated then, one obtains so called *super-linear convergence* [15]; that is convergence at a rate, that increases per iteration. This phenomenon is explained by the fact that [16], CG tends to eliminate components of error in the direction associated with extremal eigenvectors and these eigenvalues are not considered in the subsequent iterations, that is, the convergence depends upon a reduced system with much smaller condition number.

2.3.5 Biconjugate gradient stabilized (BiCGSTAB)

For solving non-symmetric linear systems, the BiCG was developed as a generalization of CG. However, operations of transpose matrix-vector multiplications are needed in BiCG because a Krylov subspace generated from the transpose matrix is used. In order to avoid calculating the transpose matrix-vector multiplications and improve the convergence rate in BiCG, efforts have been devoted to deriving more efficient methods from restructuring BiCG. In the CG method the residual vector can be regarded as the product of $r^{(0)}$ and an ith degree polynomial in A, that is [4];

$$r^{(i)} = \mathcal{P}_i(A)r^{(0)}$$

This is known as the residue polynomial. A common technique to modify the BiCG method is to define its residue polynomial by a product of two polynomial factors where one factor is the residual polynomial of BiCG and the other one is an undetermined polynomial. For example in the conjugate gradient squared CGS the undetermined polynomial is defined by same residual polynomial of BiCG that is by the square of that of BiCG. In BiCGSTAB a polynomial is selected with two term recurrence relations instead of one factor of CGS to design the residual polynomial. This choice makes the algorithm more efficient and and stable. In fact, many numerical experiments also indicated that BiCGSTAB can often run faster and its convergence behavior is smoother.[17, 12. 16].

2.4 Genetic Algorithms

Genetic algorithms which mimic natural evolutionary principles, to constitute search and optimization procedures, are different in variety of ways than classical methods. This section discusses several aspects of GAs and how they are tailored for the present problem. Before that some of the difficulties encountered in the classical approach are discussed.

2.4.1 Disadvantages of classical methods

- In the classical methods discussed so far there is some assumption made about the nature of the problem that is, for these methods to converge the coefficient matrix must have certain properties. For example Jacobi and Gauss-Seidel methods converge when the coefficient matrix is diagonally dominant.

- CG the most popular of all the methods, also assumes that A must be symmetric and positive definite.

- Because of these difficulties not every method is applicable to every problem, the knowledge of matrix properties is the main criterion for selecting a classical method. Often a lot of experience is required.

- These methods require some initial guess to start iterations Although theoretically the convergence does not depend upon the initial guess but in some cases, as we shall see later it does affect the convergence.

- These methods are not directly applicable to non-linear problems.

2.4.2 Working principles of GAs

GAs based on the Darwinian *"survival of the fittest"* principle, use the concepts of *selection, crossover or recombination* and *mutation* from genetics. The selection operator identifies good individuals which take the place of bad ones in the population. New solutions are created using the crossover operator which are then randomly picked and mutated with the mutation operator. Crossover and mutation operators can be applied to binary as well as real coded GAs. Crossover operator is the main operator which emphasizes global search and mutation is carried out for local search. Besides these operators, it has been realized that other genetic algorithm models than a simple genetic algorithm for real parameter optimization is also important. In the following we describe, a model that has been used in the present study

Minimal Generation Gap (MGG) model

This is a steady-state model, where the recombination and selection operators are intertwined. This has been modified to be computationally faster by replacing previously used roulette-wheel selection with a tournament selection operator. This model also preserves elite solution from the previous iteration. The model is as follows;

1. From the population $P(t)$, select μ parents randomly.
2. Generate λ offspring from μ parents using recombination scheme.
3. Choose two parents at random from μ chosen parents.
4. From a combined subpopulation of chosen two parents and λ offspring, choose the best two solutions and replace the chosen two parents with these solutions.

The above procedure completes one iteration of MGG model.

Recombination operator

In recent times, many researchers have been paying attention on development of different recombination operators, blend crossover (BLX), simulated binary crossover (SBX), unimodal normal distribution crossover(UNDX), simplex crossover (SPX), are commonly used. A number of other recombination operators, such as arithmetic crossover, intermediate crossover, extended crossover are similar to BLX operator (Deb,2001; Herrara et. al,1998). In the recent past, GAs with some of these recombination operators have been demonstrated to exhibit self-adaptive behavior similar to that in evolution strategy (ES) and evolution programming approaches.

Beyer and Deb (2001) argued that a recombination operator may have the following two properties:

1. Population mean decision vector should remain same before and after recombination operator.

2. Variance of the intra-member distances should increase due to the application of recombination operator.

Since the recombination operator does not use any fitness function information explicitly, the first argument is explained. The second argument comes from the realization that selection operator has a tendency to reduce the population variance. Thus, population variance must be increased by the recombination operator to preserve adequate diversity in the population. One method to preserve the population mean is to have individual recombination events preserving the mean between the participating parents and resulting offspring. This approach is called *mean-centric* recombination. The other approach would be to have individual recombination event biasing offspring to be created near the parents, but assigning each parent an equal probability of creating offspring in its neighborhood. This will also ensure the population mean of the entire offspring population is identical to that of parent population. This approach has been called *parent-centric* recombination. Based on these reasonings Parent-Centric Recombination(PCX) operator has been designed [18]. Following is the procedure to create new solutions using PCX operator.
The mean vector $\vec{g}$ of the chosen μ parents is computed. For each offspring, one parent $\vec{x}^{(p)}$ is chosen with equal probability. The direction vector $\vec{d}^{(p)} = \vec{x}^{(p)} - \vec{g}$ is calculated. Thereafter, from each of the other $(\mu - 1)$ parents perpendicular distances D_i to the line $\vec{d}^{(p)}$ are computed and their average $\bar{D}$ is found. The offspring is created as follows:

$$\vec{y} = \vec{x}_p + w_\zeta |\vec{d}^{(p)}| + \sum_{i=1 \quad i \neq p}^{\mu} w_\eta \bar{D} \vec{e}^{(i)} \tag{2.25}$$

where $\vec{e}^{(i)}$ are the $(\mu - 1)$ orthonormal bases that span the subspace perpendicular to $\vec{d}^{(p)}$. Thus, the complexity of the PCX operator to create one offspring is $\mathcal{O}(\mu)$. The parameters w_ζ and w_η are zero-mean normally distributed variables with variance σ_ζ^2 and σ_η^2 respectively.

Mutation operator

One of the most striking feature of GAs is the flexibility with which they can be modified for a particular problem. This has been done in the present work. The procedure of creating new solutions using PCX operator is very generic, it does not involve any *problem information.* This difficulty is circumvented by using any of the classical method, described above, in the place of mutation operator.

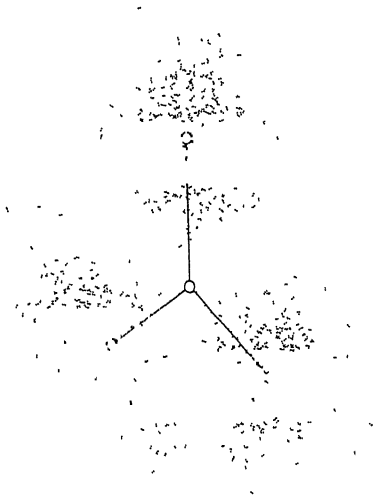


Figure 2.2: PCX operator

2.5 Modified procedure GA with a classical method

A procedure has been developed which uses MGG model, the PCX operator and classical method to solve the linear system $Ax = b$. Although most of the iterative methods minimizes the quadratic form (??), the fitness function used for GA is same as the termination criteria of classical method. That is;

$$f(x) = \frac{\|Ax - b\|}{\|b\|}. \tag{2.26}$$

This is done because classical methods require to formulate it as an optimization problem on which gradient based methods can be applied. Since gradient of the objective function as in equation (2.16) is already known that is, $(Ax - b)$ these methods do not evaluate it explicitly, which is an $\mathcal{O}(n^3)$ task otherwise. It is, as if, one is trying to maximize the function $\sin(x)$ between 0 to π and substitutes the value $\cos(x)$ whenever gradient computation is required. As GAs are not based on gradient information and the tournament selection operator of GAs requires only the difference in objective function values between two solution for comparison similar results can be obtained by using much simpler equation as in (2.26). The whole procedure can be summarized as follows:

Step 1 A population is created randomly and the best individual is identified.

Step 2 New solutions are created using the PCX operator around the best individual.

Step 3 At random, two parents are selected and from the combined subpopulation of parents and newly created solutions two best solutions are selected and they are replaced by the chosen parents.

Step 4 A fixed number of iterations of classical method are carried out on the best solution in the population.

this completes one generation of the modified procedure Here new solutions are created around the best parent always and similerly it is the best individual that is mutated in every generation. This makes the algorithm somewhat greedy but faster. There are many parameters, which are to be fine tuned according to number of variables that is the size of the grid. The most important one is the number of iterations of classical method used for the mutation operator. There is a trade off involved with respect to the cost of computation and between determinism and stochasticity as well. Secondly we can create different number of solutions in **Step 2** and number of parents to be replaced (1 or 2) in **Step 3**. Also number of parents taking part in the crossover operator can also be varied with the minimum being 3.

2.5.1 Non-linearity

It is found that non-linearity can also be handled by the same procedure. The fitness is evaluated from non-linear FD algebraic equation. But in the mutation operator, that uses classical method the residue is evaluated without updating k, as this is done in the classical method alone. This has been pointed out earlier that classical methods cannot deal with non-linear equation without linearizing it.

Chapter 3

Simulation Results

3.1 Introduction

The algorithm developed has been applied to various problems taken from 2-D heat conduction. Different sizes and types of grids as well as different boundary conditions have been tested. Non-linear equation in which the coefficient of thermal conductivity is a function of temperature has been considered as well. All problems have been solved in the physical plane itself except when the grid is in logarithmic progression, grid transformation has been applied. Governing equation of heat conduction through a solid is the familiar Laplace's equation $\nabla^2 T = 0$

Benchmarking criterion

This has been chosen carefully since it is difficult to define the term *function evaluations* in case of classical methods, results have been compared with respect to elapsed CPU time. Even this also can be questioned as it is subject to the efficiency with which an algorithm has been implemented on a computer. Moreover, the storage requirement is another matter which the elapsed CPU time do not portray. But codes for classical method as well as GA have been fairly optimized and the results are taken on a P-III processor with 128 MB RAM. Here we use a serial processor for all computing hence we note that GAs can be well parallelized for reducing computing time, a result we do not exploit in this work.

3.2 Uniform grid

The equation that governs $2-D$ heat conduction is given by Laplace's equation;

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)=0, \tag{3.1}$$

The finite difference approximation of the above equation for isotropic material, and constant k is given by;

$$T_{i,j}-\frac{1}{4}(T_{i+1,j}+T_{i-1,j}+T_{i-1,j}+T_{i,j+1}+T_{i,j-1})=0. \tag{3.2}$$

This equation shows that temperature at any point in the domain, is the average of temperature at four adjacent nodes. The adjacent nodes are located at east, west, north and south of the node in concern. This equation has been solved in the domain 0 to 1, Dirichlet as well as Neumann boundary conditions have been applied as shown in Figures 3.1 and 3.2. With Neumann B.C., three sides have been insulated and on one side temperature as a function of x coordinate $100sin(\pi x)$ has been applied.

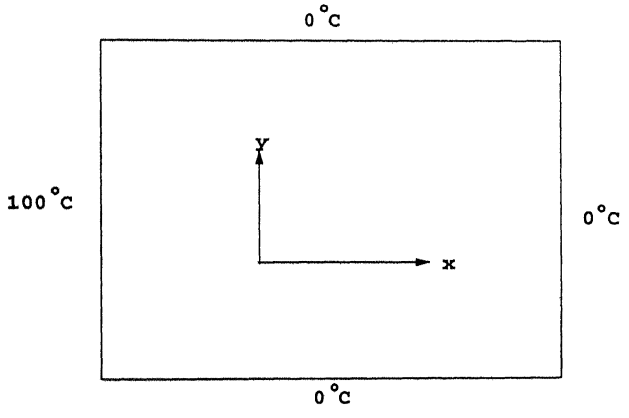


Figure 3.1: Dirichlet boundary condition

This has been done to introduce asymmetry in the coefficient matrix. The corner points where Neumann conditions exist in both the directions have been handled using the image point technique as shown in the figure 3.3, and for remaining boundaries a similar technique has been used. Using this formulation, equation (3.2) for corner point (1,1) becomes:

$$T_{1,1}-\frac{1}{2}(T_{2,1}+T_{1,2})=0, \tag{3.3}$$

which also introduces asymmetry in the coefficient matrix.

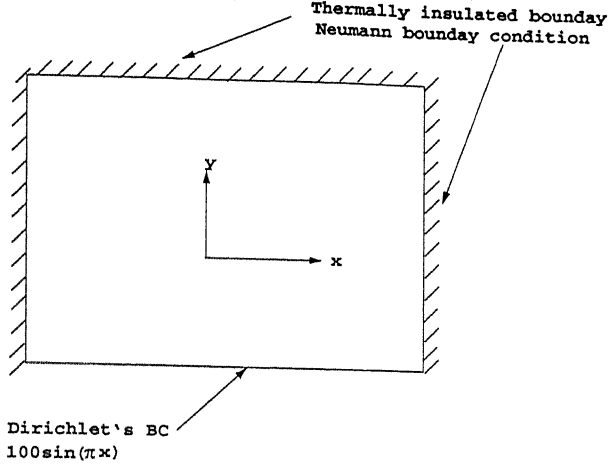


Figure 3.2: Neumann boundary condition

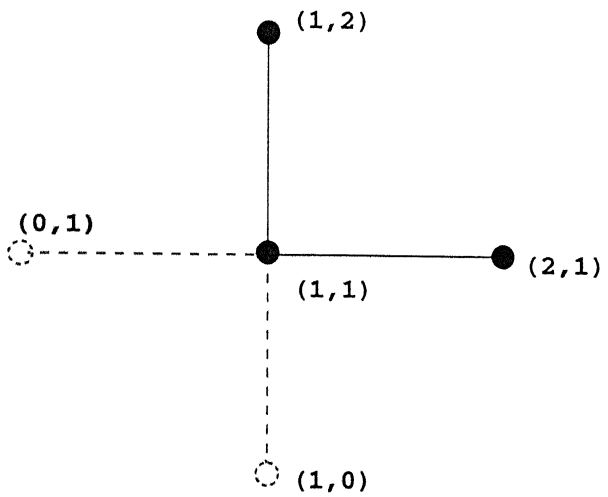


Figure 3.3: Image point technique

First the GA alone with MGG model as discussed in 2.4.2, has been applied to this problem without any mutation operator. Results are shown in Figure 3.4. Here the parameters, *kids* represents the number of new solutions created and *family*, number of parents to be replaced with good individuals. In this run, we keep function evaluations that, is number of times the residue is evaluated on x-axis, which is found to be of the order of 10^7. Following conclusions can be drawn from the results of Figure 3.4; A GA without any problem information takes a large number of function evaluations and lot of CPU time. This can be explained by the fact that the solution of the equation $Ax = b$ represents a point in n-dimensional search space, where n the size of the x vector. Here we use $n = 400$. The GA is started with a random population scattered in the search space.

This result suggested the use of a problem specific mutation operator. Different mutation operators such as the Jacobi method, the Gauss-Seidel method, the CG method and

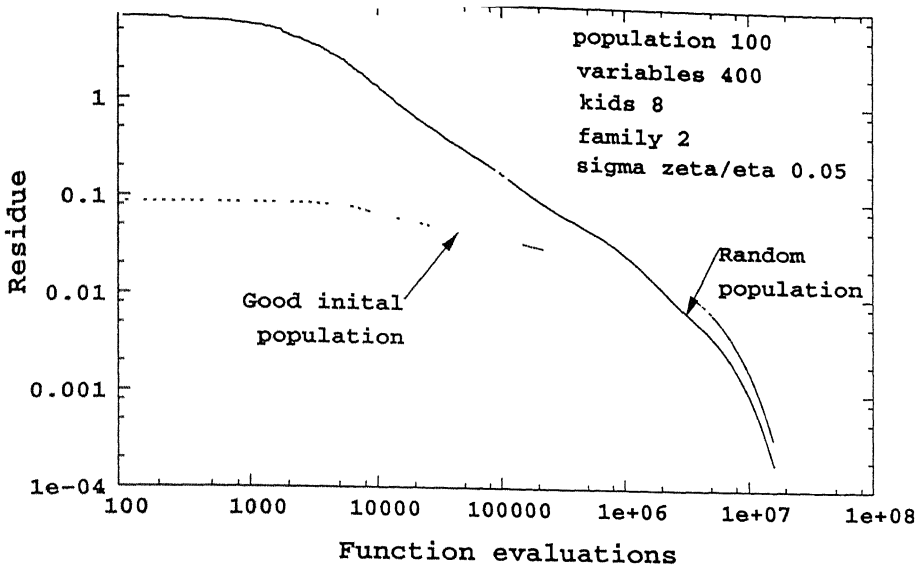


Figure 3.4: GA without any problem information for Dirichlet's B.C.

combination of them can been applied. As described earlier the mutation operator picks the best solution and applies several iterations of any of the classical method. Figures 3.5 and 3.6 show how CPU time required to reach the residue below 10^{-6} reduces with increasing the number of iterations. The mutation operator used are Gauss-Seidel method and CG method. Number of variables are 2500. Following conclusions can be drawn from these figures;

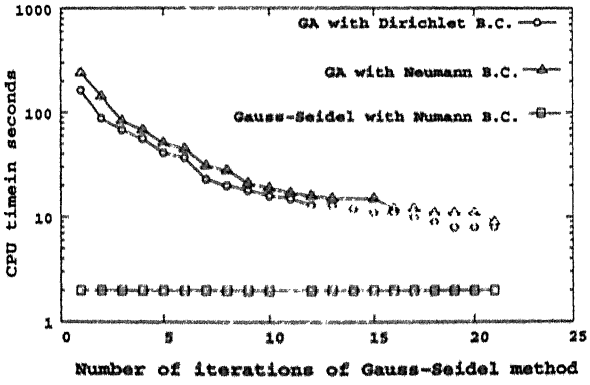


Figure 3.5: Gauss-Seidel method as the mutation operator

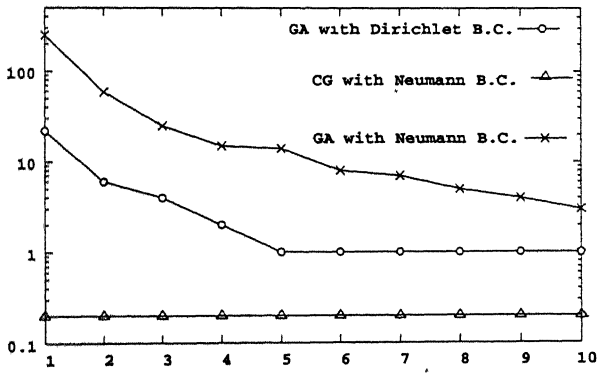


Figure 3.6: CG method as the mutation operator

- As the number of iterations are increased, the CPU time for a GA asymptotically approaches towards that of the classical method. Hence, it can be concluded that convergence of combined GA and classical method is due to classical method only, otherwise as in Figure 3.7 the curve corresponding to GA is lower than that of BiCGSTAB. In other words performance of GA with the classical method as mutation operator is

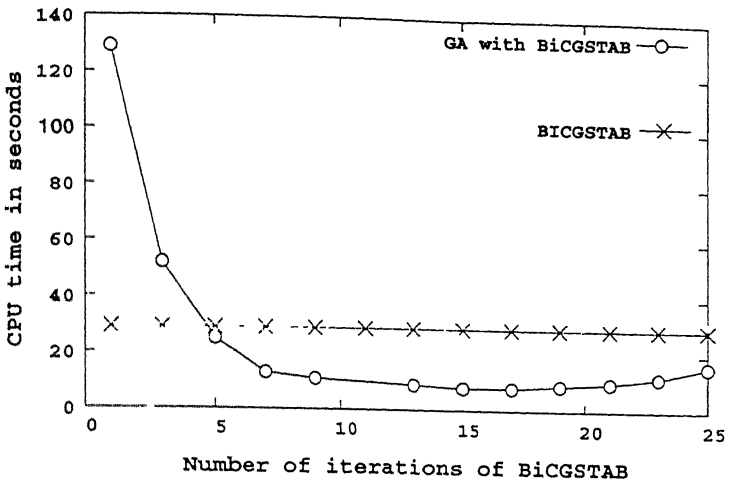


Figure 3.7: Non linear problem and BiCGSTAB method as mutation operator

even worse than classical method alone. This can be explained by the fact that the equation (3.2) gives rise to a coefficient matrix with diagonal entries unity. According to Gershgorin's "circle theorem": *Every eigenvalue of A lies in at least one of the circles $C_1, \ldots C_n$, where C_i has its center at the diagonal entry a_{ii} and its radius $r_i = \sum_{j \neq i} |a_{i,j}|$ equal to the absolute sum along the rest of the row.* The circle drawn with diagonal entry as center and sum of absolute values of rest of the entries along the row as radius is known as Gershgorin disc. Obviously if the diagonal entries are same then all the Gershgorin discs will be concentric. This will make the ratio of largest to smallest eigenvalue, that is the condition number, much lesser. Here, we are solving a system in which all the diagonal entries are unity, convergence of classical method is much faster. Hence, it is not worthwhile to use GA in its present form for such problems.

- We generate good initial population such that, each member in the population satisfies the boundary conditions. This is ensured by creating random points on a *non-uniform rational B-spline surface* (NURBS). It can be seen from the Figure 3.4 that, although initially the residue is much less but afterwards it does not matter whether we are using a random population, or a good initial population.

- Although it has been assumed in the derivation of CG method that coefficient matrix should be symmetric and positive definite but the method works well for slightly asymmetric matrices as well.

- To increase the complexity of the problem, grid stretching in the physical plane can be done so that the corresponding finite difference equation in the transformed plane will have coefficients as a function of space coordinates

3.3 Logarithmic grid

This type of grid is used when there is a very steep variation of function all-over the domain. In the physical plane, the grid can be stretched in both or at least in one coordinate direction. After suitably transforming the differential equation in the computational plane uniform spacing is obtained. In general, the Laplace's equation in the transformed orthogonal plane (ξ, η) plane is given by;

$$\frac{\partial}{\partial \xi}\left(\frac{h_2}{h_1}\frac{\partial T}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{h_1}{h_2}\frac{\partial T}{\partial \eta}\right) = 0, \tag{3 4}$$

where

$$h_1 = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2}, \quad \text{and} \quad h_2 = \sqrt{\left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2}.$$

Let us replace f by $\frac{h2}{h1}$ and g by $\frac{h1}{h2}$, then by using following finite difference scheme, we can preserve the symmetry of the coefficient matrix;

$$\frac{\partial}{\partial \xi}\left(f\frac{\partial \psi}{\partial \xi}\right)_{i,j} = \frac{1}{\triangle \xi^2}[f_{i-\frac{1}{2},j}T_{i-1,j} - (f_{i-\frac{1}{2},j} + f_{i+\frac{1}{2},j})\psi_{i,j} + f_{i+\frac{1}{2},j}T_{i+1,j}], \tag{3 5}$$

$$f_{i-\frac{1}{2},j} = \frac{1}{2}\left[f_{i-1,j} + f_{i,j}\right], \tag{3.6}$$

$$f_{i+\frac{1}{2},j} = \frac{1}{2}\left[f_{i,j} + f_{i+1,j}\right], \tag{3.7}$$

$$\frac{\partial}{\partial \eta}\left(g\frac{\partial \psi}{\partial \eta}\right)_{i,j} = \frac{1}{\triangle \eta^2}[g_{i,j-\frac{1}{2}}\psi_{i,j-1} - (g_{i,j-\frac{1}{2}} + g_{i,j+\frac{1}{2}})\psi_{i,j} + g_{i,j+\frac{1}{2}}\psi_{i,j+1}], \tag{3 8}$$

$$g_{i,j-\frac{1}{2}} = \frac{1}{2}\left[g_{i,j-1} + g_{i,j}\right], \tag{3.9}$$

$$g_{i,j+\frac{1}{2}} = \frac{1}{2}\left[g_{i,j} + g_{i,j+1}\right]. \tag{3 10}$$

The transformation to map the physical plane to the computational plane by exponential stretching in y direction is given by,

$$x = \xi \quad \text{and} \quad y = e^{\eta} - 1,$$

For this transformations; $h1 = 1$, $h2 = e^{\eta}$ and $f = e^{\eta}$, $g = e^{-\eta}$.

3.3.1 Dirichlet B.C.

Similar boundary conditions as shown in the figure 3.1 has been applied and the results are taken for different grid sizes. Figure 3.8 and 3.9 shows best GA run (out of 10) with CG and Gauss-Seidel methods. Table 3.1 summarizes results for GA runs each time starting with different random initial population. The parameters used for GA in this problem and all problems to be considered are:

population:	20
kids:	2
family:	2
σ_ζ,σ_η	0.05

where σ_ζ and σ_η are the standard deviations for the normal distribution used, in creating new solutions as discussed in section 2.4.2 The mutation operator used with GA is the CG method. These results show that GA is better than the Gauss-Seidel method but worse than the CG method. This is according to the theory [4]. The CG method is two order magnitude better in terms of CPU time this trend can be seen for grid sizes 100×100 and 150×150. But for grid size 50×50 and variables 2500 the CPU time is comparable for all the algorithms. Here in the Table 3.1 '0' indicates that the algorithm is taking less than a second to converge.

Table 3.1: Comparison for Dirichlet B.C. and Logarithmic grid

Variables	CPU time in seconds				
	GA with CG			CG	GS
	Best	Worst	Median		
2500	1	1	1	0	3
10000	21	26	24	1	55
22500	119	147	133	4	356

3.3.2 Neumann B.C.

Neumann conditions on three sides of the square, have been applied as shown in the Figure 3.2 and best GA run with CG and Gauss-Seidel are shown in Figures 3.11 and 3.12

Results for all the GA runs are summarized in the Table 3.2. These results show that when the number of variables are less that is, for grid size 50×50 and variables 2500 performance of GA is much better than the CG method. Since the CG method requires

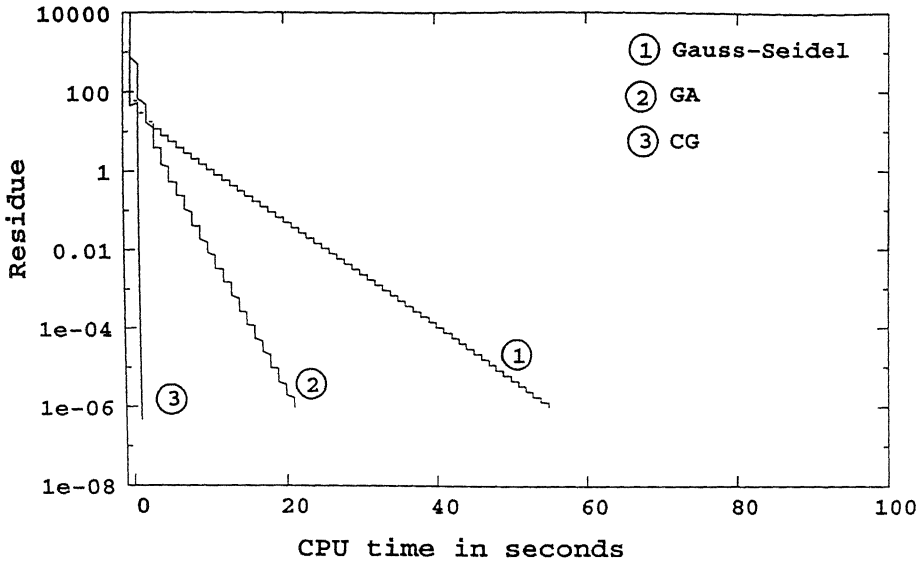


Figure 3.8: Residue history of methods using logarithmic grid, Dirichlet B C. for grid size 100×100 (10000 variables)

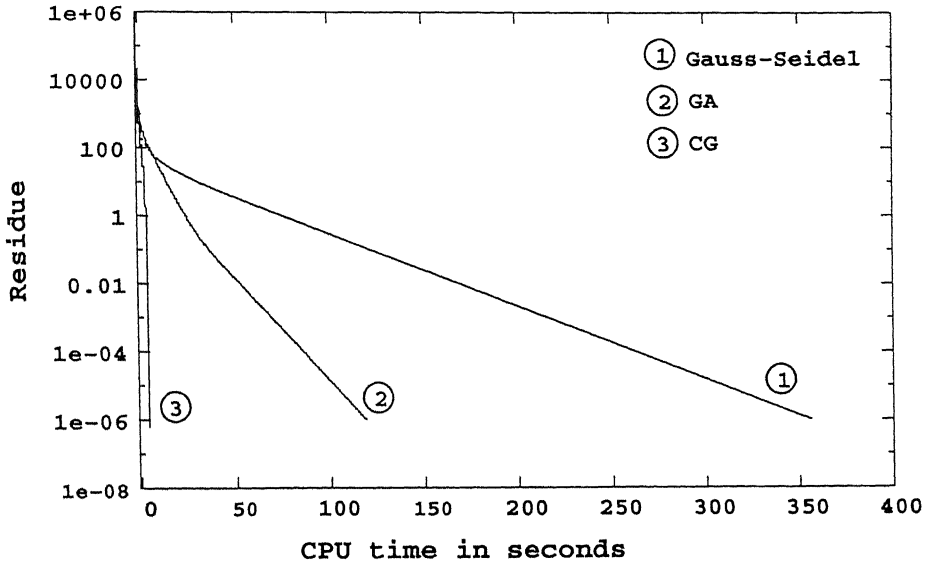


Figure 3.9: Residue history of methods using logarithmic grid, Dirichlet B.C. and grid size 150×150 (22500 variables)

some initial guess, which is generally taken as null vector, or any other vector known from previous analysis. Here, it turns out that if we take starting vector as null vector, then the convergence of the CG method is not smooth, and it takes even more CPU time than Gauss-Seidel method. But if any other suitable guess is taken, then the convergence is instantaneous. This trend is followed for grid sizes 100×100 and 150×150 but at least the convergence of CG method with null vector as the initial guess, is better than Guass-Seidel. This can be explained by the fact that as we use larger grid sizes the truncation error goes

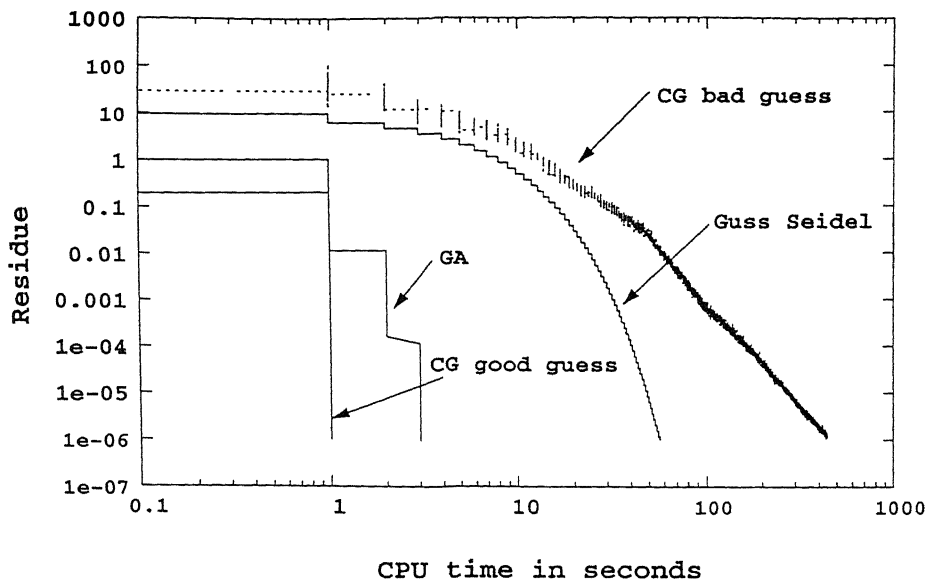


Figure 3.10: Residue history of methods using logarithmic grid, Neumann B.C. and grid size 50×50 (2500 variables)

Table 3.2. Comparison for Neumann B C. and Logarithmic grid

Variables	CPU time in seconds					
	GA with CG			CG		GS
	Best	Worst	Median	Good guess	Bad guess	
2500	3	7	5	1	445	57
10000	131	173	154	5	121	1068
22500	286	399	339	21	256	1302

to zero and the FD equation get closer to the original differential equation [1]. For GA it can be observed from Figures 3.11 and 3.12, that, for number of variables 10000 and 22500, performance of the GA is closer to the CG method with null vector as the initial guess. The final outcome of these figures is as follows;

- Though the CG method converges for slightly asymmetric matrices, but its performance is highly dependent on the initial guess.
- As the number of variables are increased performance of GA gets closer to the CG method with bad initial guess.

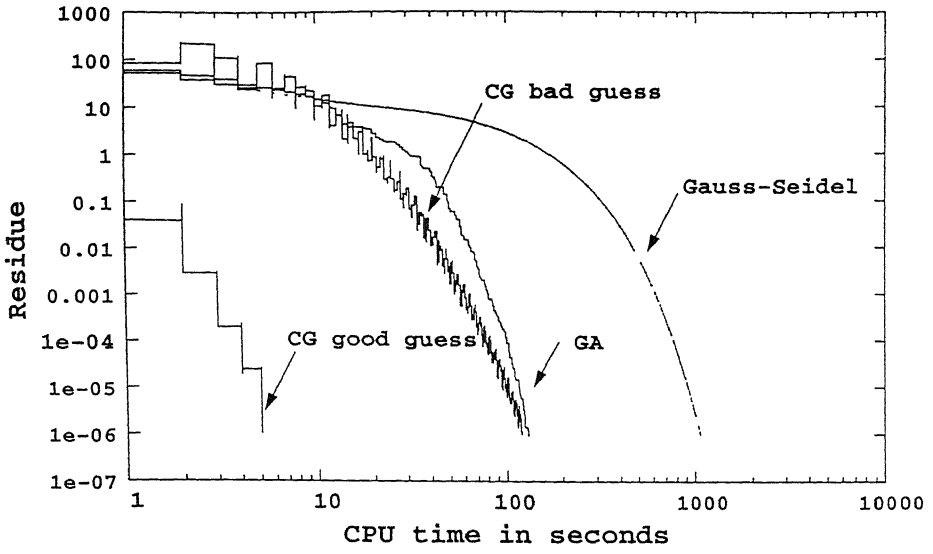


Figure 3.11: Residue history of methods using logarithmic grid, Neumann B.C. and grid size 100×100 (10000 variables)

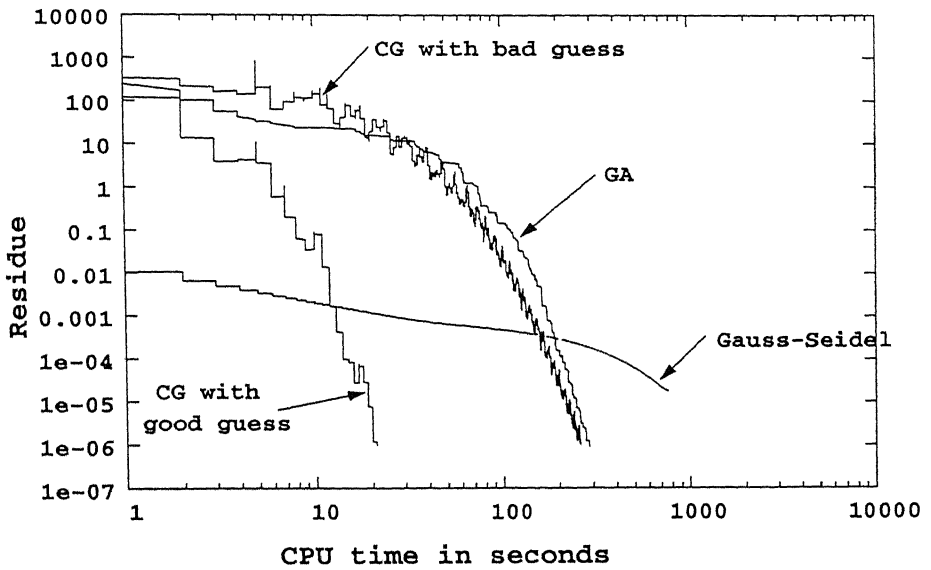


Figure 3.12: Residue history of methods using logarithmic grid, Neumann B.C. and grid size 150×150 (22500 variables)

3.4 Grid in Arithmetic Progression

Such a grid is used in the region where gradients of the unknown (for example, near the wall in boundary layer flow and/or heat transfer or heat conduction in a semi-infinite solid) are expected to be high. Therefore by making a fine grid in the region of large gradients and coarse grid in the region where the gradients are small, one can save on computer memory and execution time. This type of grid has been formed such that the distance between

nodes is in arithmetic progression and their sum is equal to one, since domain lies between 0 to 1 for both coordinate directions. Grid is stretched in only one coordinate direction that is in the x direction. Transformation of grid has not been done and differential equation has been solved in the physical plane itself. The finite difference formulation which accounts for different nodal spacing is as follows [19];

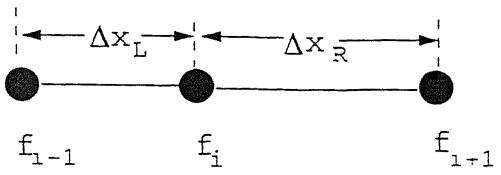


Figure 3.13: The distance between nodes is different

As shown the in Figure 3.13 for i^{th} node the distance from $(i-1)^{th}$ node is $\triangle x_L$ and from $(i+1)$th node is $\triangle x_R$. Expanding the function at $(i-1)^{th}$ node and $(i+1)^{(th)}$ node in Taylor's series and neglecting higher order terms:

$$f_{i-1} = f_i - \triangle x_L f_i' + \frac{\triangle x_L^2}{2} f_i'', \tag{3.11}$$

$$f_{i+1} = f_i + \triangle x_R f_i' + \frac{\triangle x_R^2}{2} f_i''. \tag{3.12}$$

Multiplying equation (3.11) with $\triangle x_R$ and equation (3.12) with $\triangle x_L$ and adding we get

$$f_i'' = 2\frac{\triangle x_R f_{i-1} + \triangle x_L f_{i+1} - (\triangle x_R + \triangle x_L) f_i}{\triangle x_R \triangle x_L (\triangle x_R + \triangle x_L)}. \tag{3.13}$$

Since grid is stretched in x direction only, the grid spacing in y direction is constant. If we substitute:

$$z = \frac{\triangle x_R \triangle x_L (\triangle x_R + \triangle x_L)}{\triangle y^2}, \tag{3.14}$$

Then following equation analogous to equation (3.2) is obtained.

$$2\triangle x_R T_{i-1,j} + 2\triangle x_L T_{i+1,j} - 2(\triangle x_R + \triangle x_L + z) T_{i,j} + z T_{i,j-1} + z T_{i,j+1} = 0. \tag{3.15}$$

It can be noticed that the coefficients of variables in this finite difference equation depends upon the space coordinates hence in the coefficient matrix diagonal entries corresponding to columns in the *mesh* will be different.

3.4.1 Dirichlet B.C.

Boundary conditions as shown in Figure 3.1 are applied and results are obtained for different grid sizes.

Table 3.3: Comparison for Dirichlet B.C. and Arithmetic grid

Variables	CPU time in seconds				
	GA			CG	GS
	Best	Worst	Median		
2500	2	3	3	0	1
10000	55	63	57	3	15
22500	231	347	313	4	356

It is clear from figures 3.14 and 3.15 the performance of the CG method is far better than that of GA. Table 3.3 summarizes results for 10 GA runs, which again shows that performance of GA combined with CG method is worse than Gauss-Seidel method. This case is similar to the case discussed in section 3.2. Although we have made grid in arithmetic progression the coefficient matrix is not asymmetric. Secondly even though, we have the diagonal entries of the coefficient matrix different, but due to symmetry the and diagonal dominance maintained convergence of the CG method and the Gauss-Seidel method is better than GA. An important conclusion can be drawn from this;
It is not the non-uniformity of grid, but the boundary conditions, that makes the coefficient matrix symmetric or asymmetric.

3.4.2 Neumann B.C.

As stated earlier Neumann conditions increase the condition number of the matrix as well as it makes the matrix asymmetric Since the formulation of section 3 4 the equations are solved in the physical plane itself without preserving the symmetry, in this problem CG method as well as Gauss-Seidel method does not converge at all. The mutation operator adopted for this problem is BiCGSTAB. This method works well for non-symmetric matrices as described in section 2.3.5. Boundary conditions applied for this problem are slightly different, and shown in the Figure 3.16. In this problem the mutation operator used is BiCGSTAB. Convergence of CG and best run of GA are shown in the figures 3.17 and 3.18. Table 3.4 summarizes results for 10 different GA runs.

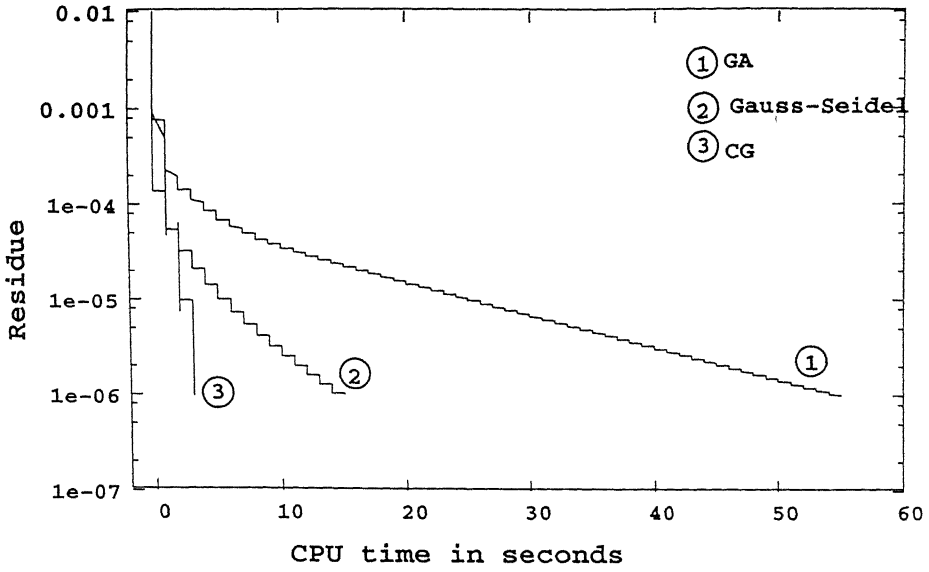


Figure 3.14: Residue history of methods using arithmetic grid, Dirichlet B.C. and grid size 100×100 (10000 variables)

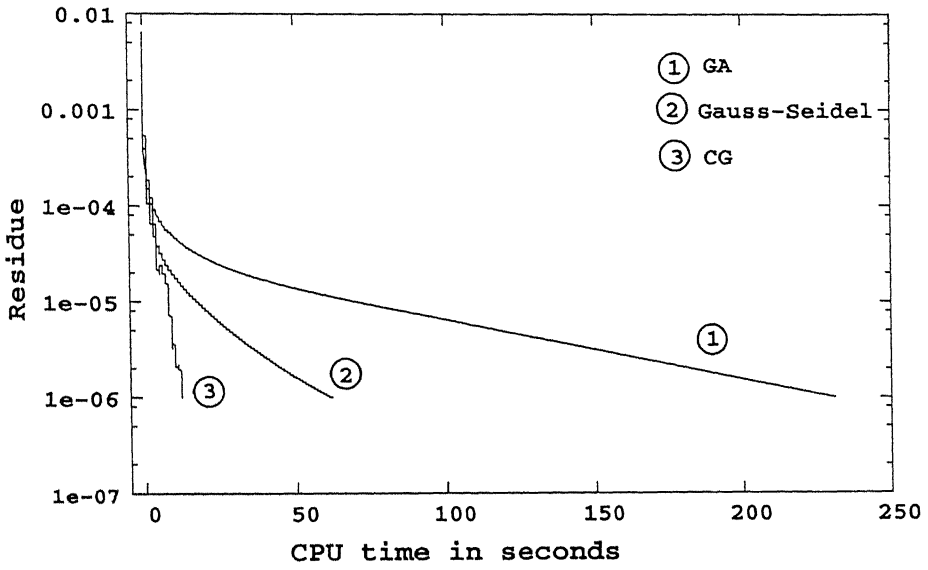


Figure 3.15: Residue history of methods using arithmetic grid, Dirichlet B.C. and grid size 150×150 (22500 variables)

3.5 Non-linear problem

Non linearity is introduced by assuming coefficient of thermal conductivity as a linear function of temperature.

$$k = k_o(1 + 3\dot{T}) \tag{3.16}$$

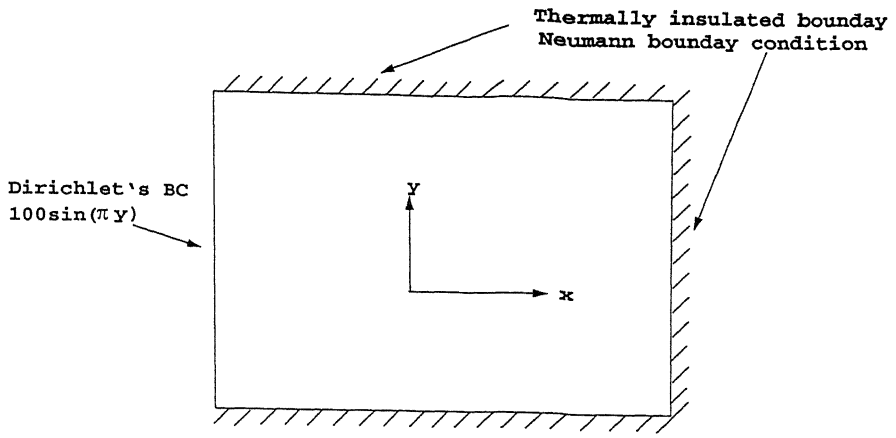


Figure 3.16: Domain for the problem with arithmetic grid and Neumann B.C.

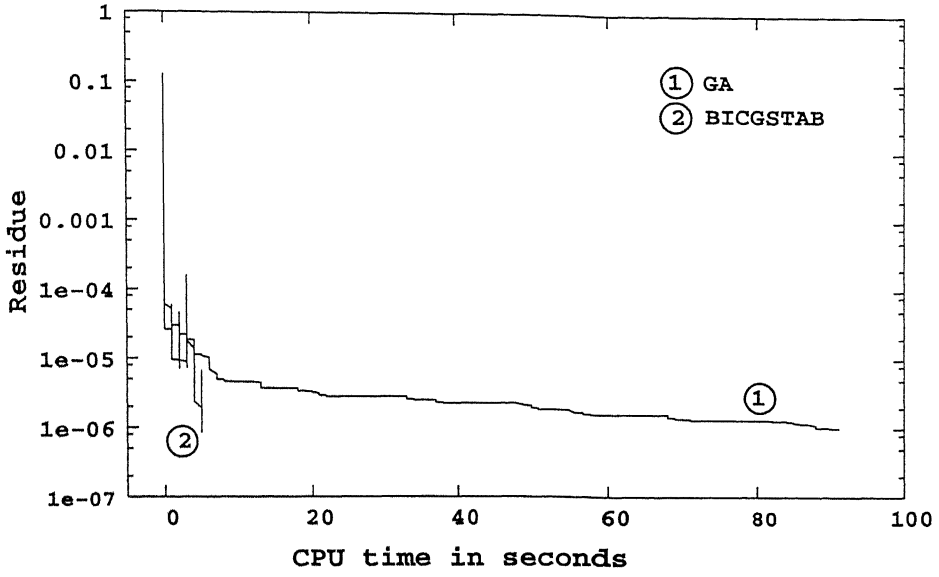


Figure 3.17: Residue history of methods using arithmetic grid, Neumann B.C. and grid size 100×100 (10000 variables)

where β is the parameter by which non-linearity can be controlled. β can take negative values. With this assumption the differential equation

$$\frac{\partial}{\partial x}\left(k(T)\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k(T)\frac{\partial T}{\partial y}\right)=0, \tag{3.17}$$

can also be expressed as,

$$k\frac{\partial^2 T}{\partial x^2}+k\frac{\partial^2 T}{\partial y^2}+\frac{\partial T}{\partial x}\frac{\partial k}{\partial x}+\frac{\partial T}{\partial y}\frac{\partial k}{\partial y}=0, \tag{3.18}$$

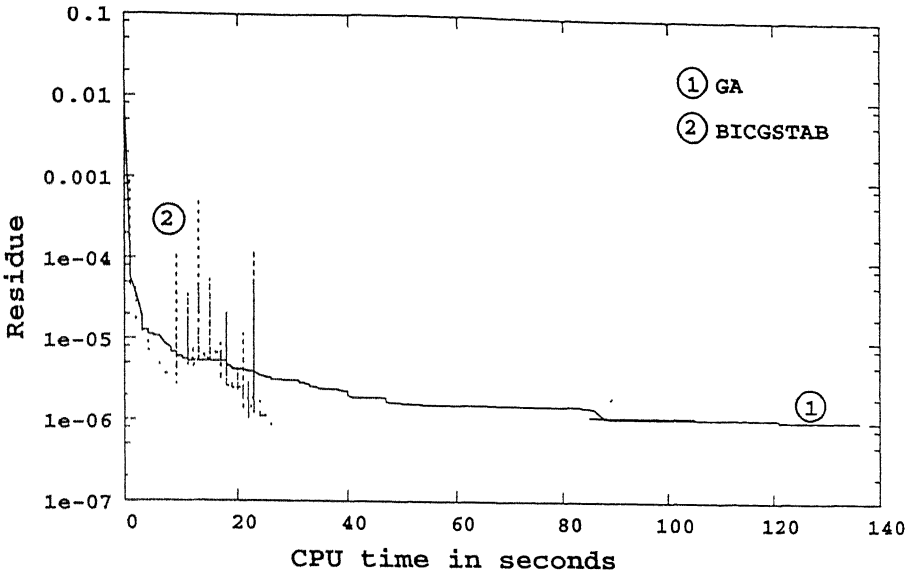


Figure 3.18: Residue history of methods using arithmetic grid, Neumann B C. and grid size 150×150 (22500 variables)

Table 3.4: Comparison for Neumann B.C. and Arithmetic grid

Variables	CPU time in seconds			
	GA			BiCGSTAB
	Best	Worst	Median	
2500	2	3	3	0
10000	91	357	128	5
22500	136	195	511	26

The finite difference equation of above differential equation is [19],

$$A_{i,j}(T_{i+1,j} + T_{i-1}j) + B_{i,j}(T_{i,j+1} + T_{i,j-1}) - C_{i,j}T_{i,j} = 0, \tag{3.19}$$

where

$$A_{i,j} = k_{i,j} + \frac{k_{i+1,j} - k_{i-1,j}}{4}$$

$$B_{i,j} = k_{i,j} + \frac{k_{i,j+1} - k_{i,j-1}}{4},$$

$$C_{i,j} = -4k_{i,j}.$$

$$k_{i,j} = k_o(1 + \beta T_{i,j}).$$

In practice such type of equation is solved by *linearizing* it. First, an initial temperature distribution is assumed and then distribution of coefficient of thermal conductivity is cal-

culated from equation (3.16). Then one or more iterations of the method are carried out and again the distribution for k is updated. Convergence is achieved when the residue before and after updation is not much different. But with GA there is no need of linearizing the equation, as GAs do not assume anything about the nature of equations to be solved. Therefore, whenever residue is calculated the current distribution of k is always used. The mutation operator is used as earlier but k is not updated when iterations of classical method are carried out, because if updation is carried out much frequently classical method may diverge. With Gauss-Seidel method updation can be carried out in each iteration but with BiCGSTAB, it can not be. Note that CG method can not handle this system of equations as the coefficient matrix is not symmetric The Boundary conditions used are Dirichlet's one with uniform grid as shown in the Figure 3 1 and negative as well as positive values of β are tested Figures 3 19 and 3.20 show convergence of GA with Gauss-Seidel and BiCGSTAB as mutation operator with value of β taken to be -0.001. Figures 3.21 and 3.22 are with β=0.05.

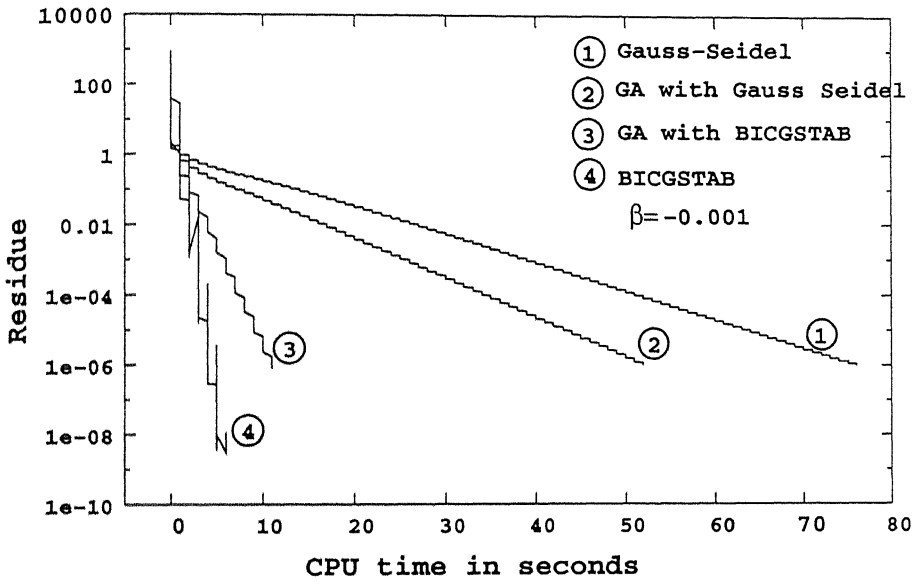


Figure 3.19: Residue history of methods, for non-linear problem and grid size 100×100 (10000 variables)

In these runs k is updated in each iteration for Gauss-Seidel method. But for BiCGSTAB k is updated after few iterations. Results for negative β shows that when Gauss-Seidel method is used as a mutation operator, then the combined GA works better than the Gauss-Seidel method alone. This can be seen in Figures 3.19 and 3.20. But when BiCGSTAB is used as the mutation operator, performance of such a GA is poor than the BiCGSTAB alone. If the value of β is reduced further then both the classical methods, BiCGSTAB as well as Gauss-Seidel method diverge. This may be due to introduction of saddle point in

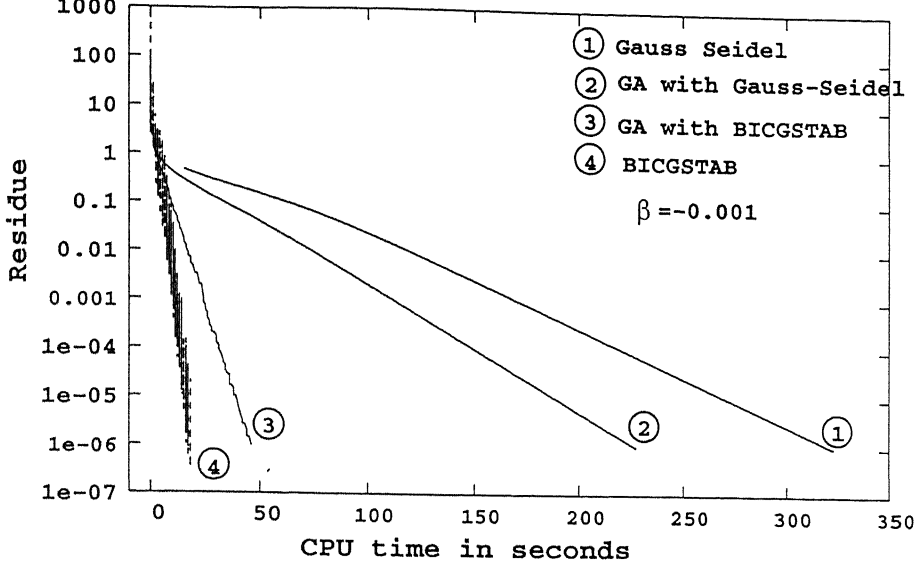


Figure 3.20: Residue history of methods, for non-linear problem and grid size 150×150 (22500 variables)

Table 3.5: Non linear problem with β=−0.001

Variables	CPU time in seconds					
	GA with BiCGSTAB			GA with GS	CG	GS
	Best	Worst	Median			
2500	0	1	0	2	0	3
10000	11	12	12	52	6	76
22500	46	61	47	227	18	323

the actual solution due to extreme value of β.

Table 3.6: Non linear problem β=0.05

Variables	CPU time in seconds				
	GA with BiCGSTAB			GA with GS	CG
	Best	Worst	Median		
2500	0	2	1	1	1
10000	15	18	17	30	30
22500	65	101	87	123	94

With β=0.05 Gauss-Seidel method do not converge if k is updated in each iteration. Even if k is updated after certain number of iterations it is difficult to obtain convergence. When Gauss-Seidel method is used as a mutation operator in GA k is not updated at all, and iterations are carried out as if we are solving a linear equation. With this strategy

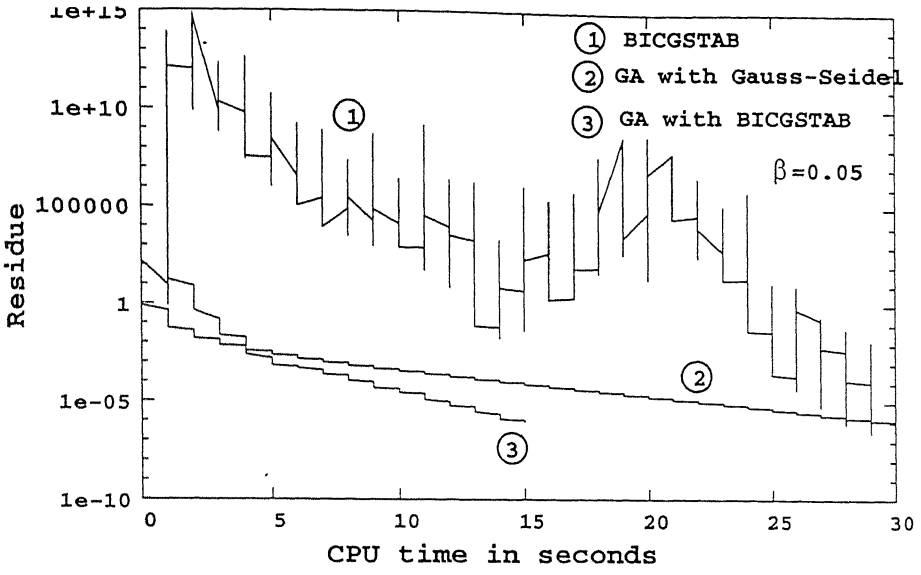


Figure 3.21: Residue history of methods, for non-linear problem and grid size 100×100 (10000 variables)

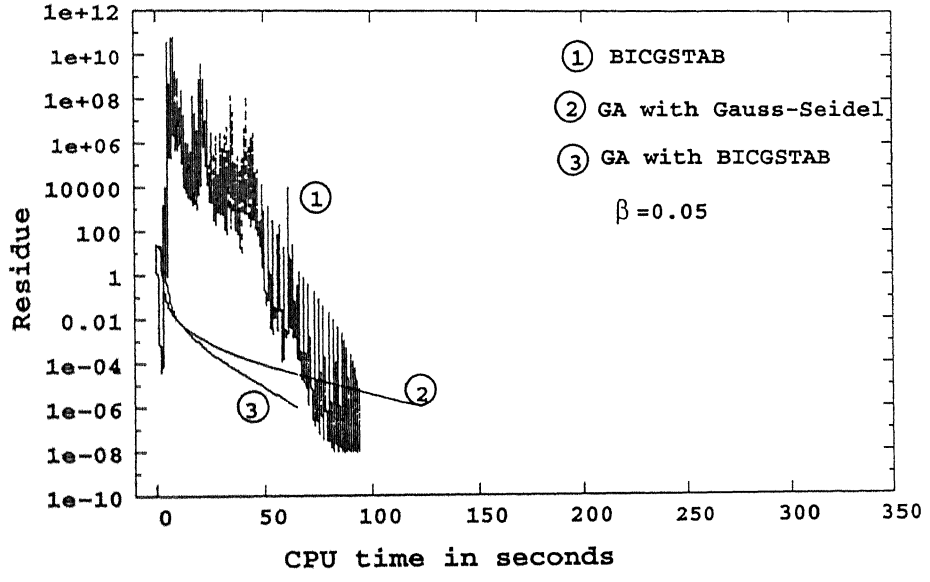


Figure 3.22: Residue history of methods, for non-linear problem and grid size 150×150 (22500 variables)

we can obtain convergence. When BiCGSTAB is used as the mutation operator then the convergence history is better than that of BiCGSTAB, and also GA takes less CPU time than the BiCGSTAB alone. Table 3.6 shows the results for positive β, results of only the best run of GA with Gauss-Seidel are included. Similarly Table 3.5 shows results for negative β. Finally we can summarize the results of GA applied to a non-linear problem

- Since GA does not involve any information about the nature of objective function it can handle a non-linear equation without linearizing it.

- We have found that the Gauss-Seidel method alone is not converging with β=0.05 but when it is used with a GA, the combined algorithm has shown better convergence.
- Whenever the distribution of coefficient of thermal conductivity is updated in case of BiCGSTAB the residue increases. But when BiCGSTAB is used as the mutation operator, smooth convergence is obtained. GA does not allow the residue to go worse since an elitist model is used in the GA. Elitism ensures the preservation of best solution. This is the reason for better performance of GA.

Chapter 4

Conclusion and Further Research

4.1 Concluding remarks

The algorithm proposed in the present work have been applied to various problem from 2-D heat conduction. Though the test problems chosen are from a narrow class of elliptic equations and boundary conditions but still some general comments can be made regarding the applicability of the algorithm.

- We have used uniform and non-uniform grid with Neumann and Dirichlet B.C.. It turns out that with Dirichlet boundary conditions the coefficient matrix is always is symmetric irrespective of the type of grid. Hence, the CG method performs better, than the GA with CG method itself as the mutation operator. Such a GA is found to perform better than Gauss-Seidel method alone, but this may be due to faster convergence of the CG method than the Gauss-Seidel. It can be also stated that contrary to the assumption made, in the derivation of the CG method, it converges for slightly asymmetric matrices as well.

- When Neumann conditions are used with logarithmic grid convergence of CG method largely depends upon the initial guess. But convergence history of GA in all the problems tackled is more or less consistent. This difference is due to population based approach of GA, starting with random initial population rules out any such dependencies.

- With arithmetic grid and Neumann conditions, the CG method does not converge at all, there fore we have used BiCGSTAB. Convergence of BiCGSTAB is not smooth but it performs much better than GA with BiCGSTAB itself as the mutation operator.

- In all the problems it has been found that when the number of variables are less, that is with grid size 50×50 CPU time for GA is comparable to that of the method used

for mutation operator. For larger grid sizes (100×100 and 150×150) GA takes order of magnitude more time than the classical method.

- For the non-linear problem and positive β the convergence of GA with BiCGSTAB as the mutation operator is found to be the best. Gauss-Seidel method does not converge on its own but if updation of k is not at all performed when it is used as the mutation operator with GA, the combined procedure converges well. But when β is negative the convergence of GA is not better than BiCGSTAB but atleast they are of the same order of magnitude.

- It is not worthwhile to use GA when the boundary conditions are Dirichlet and the coefficient matrix is symmetric.

- When the coefficient matrix is slightly asymmetric and number of variables are large then performance of GA is found to be equivalent to the classical method used as the mutation operator in the GA. Contrary to this, if the number of variables is less (of the order of 10^3 then in all the cases discussed CPU time taken for GA is of the same order as the classical method.

- Finally we can state that the developed GA is found to perform better in following cases

  - For boundary conditions which introduce asymmetry in the coefficient matrix and for cases when the number of variables are small, (of the order of 10^3).
  - For non-linear problems where the governing equation is non-linear due to temperature dependent properties

4.2 Limitations of the algorithm and suggestions for further studies

- In this work an algorithm has been developed to solve linear systems that has been applied to for 2-D heat conduction problems with Dirichlet and Neumann boundary conditions however Robin conditions (convective) and radiation boundary conditions can also be tested.

- Similarly non-linearity in the differential equation is tackled, further non-linearity due to boundary conditions and due to geometry can be tried.

- In this study only elliptic partial differential equations are dealt with. As stated earlier the crossover operators possess self adaptive behavior. This capability can be

utilized to solve parabolic equations. At each time step the objective function will change slightly and a little diversity left in the population will help in finding the new optimum solution.

- The present algorithm largely depends upon the convergence of the classical method itself. This is because GAs without the mutation operator cannot handle such a large number of variables. It is mainly due to generic nature of the PCX operator. A new crossover operator can be designed which will include problem information implicitly. Such a crossover operator would eliminate any need for the mutation operator and hence dependencies on classical method. Such a GA may be found to more cost-effective in terms of CPU time.

- Finally GAs can treat solution of largely sparse linear systems in general, to solve it efficiently and effectively, as this system occur in almost all computational areas such as networks, finite elements, computational fluid dynamics, signal processing etc

Bibliography

[1] Anderson John D. Jr. *Computational Fluid Dynamics The Basics with Applications* McGraw-Hill, New York, 105-118 (1995)

[2] Householder A.S. *Principles of numerical analysis.* McGraw-Hill, New York, (1951).

[3] Bodwig, E. *Matrix Calculus.* Wiley (Interscience) New York, (1956).

[4] Barret R. and Berry M. and Chan F T. el. al *Templates for the solution of iterative linear syetems; building blocks for iterative methods* , SIAM 6-11 (1994)

[5] Ames F. William *Numerical Methods for Partial differential equations. third edition.* Academic Press, Inc, 144-146 (1994).

[6] Hestenes M.R. and Steifel E. *Methods of conjugate gradients for solving linear systems.* In J.Res. Nat. Bur. Stand., 49, 409-436 (1952)

[7] Shewchuk J.R. *An introduction to the conjugate gradient method without the agonizing pain.* School of comp. sci. Carnegie Mallon University, (1994)

[8] Stein P., and Rosenberg, R.L. J. London Math. Soc., 23,111, (1948).

[9] van der Vorst H. A. *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems,* SIAM J. Sci. Statis. Comput., 13 631-644 (1992).

[10] Golub G. and O' Leary *Some history of conjugate gradient and Lanczos methods,* SIAM Rev.,31 50-103 (1989).

[11] Kaniel S. *Estimates for some computational techniques in linear algebra,* Mathematics of computation, 20, 369-378, (1966).

[12] van der Vorst H. A. and Driessen M. *Bi-CGSTAB in semiconductor modelling* , In Simulation of semicondutor Devices and Processes 4, W. Fichtner, ed. Hartung-Gorre, Zurich, 45-54 (1991).

[13] Strang Gilbert *Linear algebra and its applicatıons* , Saunders cllege publishing Orlando, 386-387, (1986).

[14] Malik G.M. and Jennings A. *The solution of sparse linear equatıons by the conjugate gradient method*, J. Num. methods in Engg., vol.12, 141-158, (1978)

[15] Concus P. and Golub G. and D O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, In Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, 309-332 (1976).

[16] van der Sluis A. and van der Vorst H, *The rate of convergence gradients*. Numer. Math., 48, 543-560, (1986).

[17] Shao-Liang Zhang *Generalized product-type methods based on Bi-CG for solvıng nonsymmetric linear systems*. SIAM J. Sci. Comput. 18, 2 537-551 (1997).

[18] Deb K. and Joshi D. and Anand A. *Real-Coded Evolutionary Algorithm wıth Parent-Centric Recombination*, KanGAL report No. 2001003 Kanpur Genetic Algorithms Laboratory(kanGAL) (2001).

[19] Ghoshdastidar P.S. *Computer simulation of flow and heat transfer*, Tata McGraw-Hill publishing company ltd. New Delhi, 24-27.

[20] Deb K. *Multi objective optimization using evolutionary algorithms* John Wiley and Sons, (2000).

A 143461

Date Slip

The book is to be returned on the date last stamped.

.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....